

Complexity analysis of quantum walk based search algorithms

B L Douglas and J B Wang

School of Physics, The University of Western Australia, 6009, Perth, Australia.

Abstract

We present several families of graphs that allow both efficient quantum walk implementations and efficient quantum walk based search algorithms. For these graphs, we construct quantum circuits that explicitly implement the full quantum walk search algorithm, without reference to a ‘black box’ oracle. These circuits provide a practically implementable method to explore quantum walk based search algorithms with the aim of eventual real-world applications. We also provide a numerical analysis of a quantum walk based search along a twisted toroid family of graphs, which requires $O(\sqrt{n} \log(n))$ elementary 2-qubit quantum gate operations to find a marked node.

I. INTRODUCTION

Search algorithms employing quantum walks have been the focus of several recent studies (see, for example, [1–7]). Along certain types of graphs, quantum search algorithms have been shown to yield a quadratic speedup over their classical equivalents, relative to a fixed oracle [8–10]. These types of graphs include sparse graphs with efficiently computable neighbours [11], and highly symmetric graphs such as complete graphs, hypercycles and hypercubes [7, 12].

Previous studies of quantum walk based search algorithms have focussed on a computational complexity comparison between the quantum search and the best possible classical search. These search algorithms are usually formulated relative to an oracle, a black box that accepts some input, and returns some corresponding output. A comparison is made as to the relative number of queries to the fixed oracle needed to complete the search. As such, there has been no need to consider the resource requirements of the oracle itself.

However, an efficient practical implementation of such a search algorithm would require an efficiently implementable oracle. Specifically, it requires the ability to efficiently perform steps of the quantum walk along the graph. So far efficient implementation of quantum walks has been shown to be possible only along some graphs from a certain type, whose global structure is completely characterised by a small number of parameters. Several graphs belonging to this category were considered in [12, 13], and shown to be amenable to exact, efficient quantum circuits implementing quantum walks along them.

Note that there are two differing notions of efficiency considered here. We consider an efficient quantum search algorithm to be one which exhibits at least a quadratic speedup over the best possible classical search, in some oracular setting. In the case of implementing a quantum walk along a graph on n vertices, we use the term efficient to mean that a step of the walk can be implemented using $O(\log(n))$ elementary 2-qubit gates. Which of these two situations we are referring to when we mention efficiency should be clear from the context.

In this paper we present examples of families of graphs for which both notions of efficiency hold, that is they yield both efficient quantum walk implementations and efficient quantum walk based search algorithms, without reference to an oracle. For these graphs, we construct quantum circuits that implement a step of the quantum walk (corresponding to an efficient search algorithm) using $O(\log(n))$ 2-qubit gates, for graphs on n nodes. By specifying

explicitly the method by which the walk is implemented, without reference to an oracle, we ensure there are no hidden resource requirements, and as such provide a practically implementable method to experimentally test these search algorithms.

II. QUANTUM WALK BASED SEARCH

Quantum walks can be thought of as the quantum analogues of classical random walks. They are a unitary process, and can be naturally implemented by quantum systems (see, for example, references in [14, 15]. Both discrete time [16, 17] and continuous time [18, 19] quantum walks can be considered, both of which have yielded several quantum algorithms [1–4, 20–24]. Here we consider only discrete time quantum walks. This walk consists of a unitary operator $U = SC$, acting on the state space, where S and C are termed the shifting and coin operators respectively.

Consider a discrete-time quantum walk along a general undirected graph $G(V, E)$, with vertex set $V = \{v_1, v_2, v_3, \dots\}$, and edge set $E = \{(v_i, v_j), (v_k, v_l), \dots\}$, being unordered pairs connecting the vertices. The quantum walk acts on an extended position space, in which each node v_i , having d_i edges, is split into d_i states, alternatively termed sub-nodes or coin positions. This space then consists of states $|v_i, a_i\rangle$, where $1 \leq a_i \leq d_i$. The shifting operator acts on this extended position space, with its action defined by:

$$S|v_i, a_i\rangle = |v_j, a_j\rangle, \text{ and } S^2 = \mathbf{I}$$

for some $v_j \in V$ with valency d_j , such that $(v_i, v_j) \in E$, and $1 \leq a_j \leq d_j$. The coin operator comprises a group of unitary operators, or a set of coins, which independently mix the probability amplitudes associated with the group of sub-nodes of a given node. For example, given a vertex v_i with valency d_i , the coin can be represented by a unitary $(d_i \times d_i)$ matrix. In the cases we consider here, the coin matrix is always constrained to be symmetric. Berry *et al.* [25] has recently developed a software package *qwViz*, which provides an interactive visualisation of the time-evolution of quantum walks on arbitrarily complex graphs.

One particular algorithmic application of quantum walks is that of searching for a marked node in a graph, and has been the focus of several recent studies (see, for example, [1–7]). These search algorithms generally involve starting in an equal superposition of all possible

states, and marking one or more nodes of the graph. The aim is to evolve the state by successive steps of a quantum walk, until the probability to measure the walk at a marked node is sufficiently high, at which point the state of the walk is measured.

In the following section, we present quantum circuits that implement the full quantum walk based search algorithm. In this situation the efficiency is not just measured relative to the best possible classical search in an oracular setting. Instead, since in previous quantum search studies [1–7] the oracle is queried once per step of the walk, generally taking the place of the coin or shifting operators of the walk, we will analyse the complexity of these examples based on the number of steps of the quantum walk required, together with the total number of elementary 2-qubit quantum gates required to implement the walk. Explicit circuits implementing the quantum walk based search will be provided for each family of graphs.

III. EXAMPLES

A. Hypercube

Firstly we consider the n -dimensional hypercube, with $N = 2^n$ nodes, each of degree n . The graph for $n = 5$ is shown in Fig. 1. Shenvi *et al.* [2] demonstrated an efficient quantum walk based search algorithm for finding marked nodes on the hypercube. Their algorithm operates quadratically faster than classical methods, requiring $O(\sqrt{N})$ calls to an oracle that effectively acts as a coin operator which is biased only relative to the marked node during each step of the quantum walk.

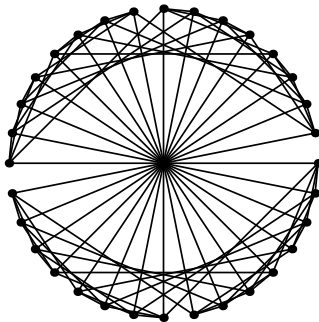


FIG. 1: The 5 dimensional hypercube.

Here we show that this quantum walk oracle can be implemented efficiently, due to the symmetric nature of the graph, requiring $O(\log(N))$ elementary 2-qubit gates per call. The coin operator C is defined as $C = G \otimes (\mathbf{I} - |x\rangle\langle x|) + C' |x\rangle\langle x|$, where C' is the perturbed coin acting on the marked node x (for the examples presented here, $C' = -\mathbf{I}$). G is the Grover operator, defined on d dimensions by $G_{i,j} = \frac{2}{d} - \delta_{i,j}$. The shifting operator has its standard definition, provided above, such that for $v_i, v_j \in V$, $S|v_i, a_i\rangle = |v_j, a_j\rangle$, where $S^2 = \mathbf{I}$ and $1 \leq a_i, a_j \leq n$.

An example quantum circuit implementing a step ($U = SC$) of the walk along the n -dimensional hypercube and biased towards the marked node is shown in Fig. 2. Here the G operator represents the Grover operator defined above, and the π operator (also known as the negative identity operator) represents a π phase change applied to each qubit it acts upon. This circuit requires n ‘node’ qubits, with each bit string representing a node of the hypercube, with the natural ordering on the hypercube, such that two nodes are connected if and only if their bit strings have a Hamming distance of 1. $\log(n)$ ‘subnode’ qubits are required (rounded up to the nearest integer), with the first n bit strings representing the n edges of each node.

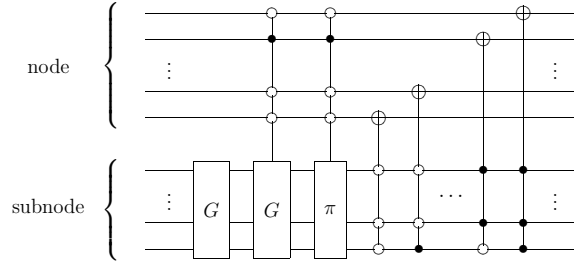


FIG. 2: Quantum circuit implementing a step of the quantum search along the n -dimensional hypercube, on 2^n nodes. The circuit contains n ‘node’ qubits and $\log(n)$ ‘subnode’ qubits, and the marked node is represented by the state $|01 \cdots 00\rangle$.

This circuit provides an efficient implementation, in that $O(\log(N))$ elementary 2-qubit gates are required per step. The Grover operator G acting on the $\log(n)$ subnode qubits requires $O(\log(n))$ 2-qubit gates [26]. The generalized conditional NOT operations are described in [27], in which it is shown that a generalized C^n NOT gate, performing a NOT operation conditionally based on the state of n other nodes, can be implemented using $O(n)$ 2-qubit gates together with $O(n)$ auxiliary qubits. Performing both these and the condi-

tional G and π operations requires $O(n)$ 2-qubit gates and an additional $O(n)$ auxiliary qubits (see, for example, [26, 27]).

Hence the circuit of Fig. 2, performing a step of the walk implementing a quantum search along the hypercube on n -dimensions (with $N = 2^n$ nodes), can be implemented using $O(\log(N))$ qubits and $O(\log(N))$ 2-qubit gates. This circuit is a representation of a single step of the search algorithm of [2], with the explicit action of the oracle included in the circuit. For the 5-dimensional hypercube, given an initial equal superposition across all states, the resulting probability distribution against number of steps is given in Fig. 3. Hence as in [2], after $O(\sqrt{N})$ steps of the walk (or equivalently $O(\sqrt{N})$ repetitions of the circuit of Fig. 2, each requiring $O(\log(N))$ elementary quantum gates), the walk will be found at the marked node with sufficiently high probability.

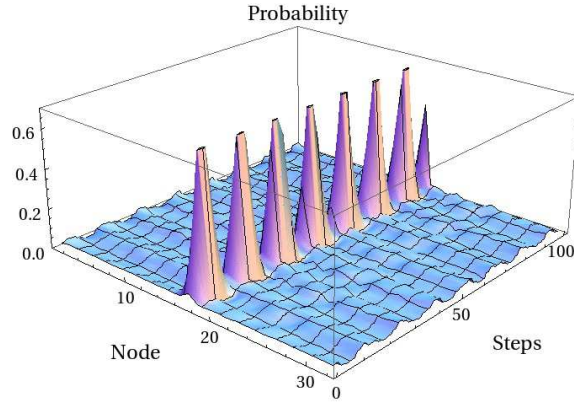


FIG. 3: Probability distribution along the hypercube against the number of walking steps.

B. Complete graph

Similar results pertain to the complete graph on $N = 2^n$ nodes. Here we consider each node to possess a self loop, so that each node has degree N . Combining the circuits of [12] for complete graphs with 2^n nodes (which efficiently implements a step of the walk along a complete graph) with the search algorithm of [7] yields a complete implementation of the search for the marked node in such graphs. Fig. 4 presents this implementation, for a single step of the walk. The shifting and coin operations are applied to the topology of the complete graph exactly as with the hypercube. As in the hypercube example above, for a complete graph on $N = 2^n$ nodes, $O(n)$ node qubits are required. In this case no

particular mapping between the bit strings and the nodes is necessary. Since each node has degree N , $O(n)$ subnode qubits are also required, together with $O(n)$ auxiliary qubits to implement the $C^n G$ and $C^n \pi$ gates. Hence $O(n) = O(\log(N))$ elementary 2-qubit gates are again sufficient to implement a step of the walk along this family of graphs. The marked node is found with sufficiently high probability after $O(\sqrt{N})$ steps (see [7]), hence in total it requires $O(\sqrt{N} \log(N))$ 2-qubit quantum gates to locate the marked node.

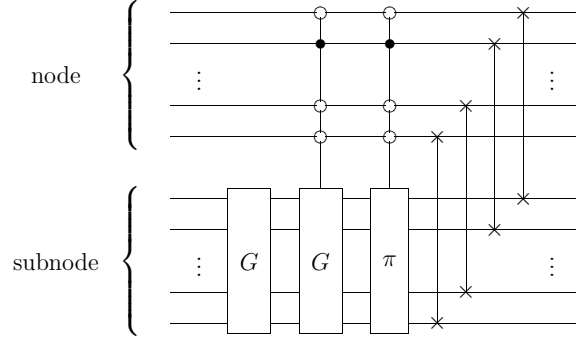


FIG. 4: Quantum circuit implementing a step of the quantum search along the complete graph on 2^n nodes. The circuit contains n ‘node’ qubits and n ‘subnode’ qubits, and the marked node is represented by the state $|01 \cdots 00\rangle$.

C. Twisted toroid

Following on from the discussion of the twisted toroid in [12], we considered the possibility of marked node searching along this family of graphs. Formally, a graph in this family of dimension $n \times m$ is constructed by taking an $n \times m$ grid, and associating the endpoints with each other as in the construction of a simple toroid, with the modification : $\{1, j\} \leftrightarrow \{n, (j+1) \bmod m\}$ and $\{i, 1\} \leftrightarrow \{(i+1) \bmod n, m\}$, where ‘ \leftrightarrow ’ denotes an edge. Here $\{i, j\}$ refers to the corresponding point on the grid, where $1 \leq i \leq n, 1 \leq j \leq m$. Fig. 5 shows a member of this family of graphs, of dimension 10×10 .

This family of graphs is highly symmetric, and can be completely characterized by a small number of parameters (growing logarithmically with the size of the graph). Given this symmetry, it is a good candidate for a graph on which both quantum search and a direct implementation of quantum walks are efficient. We previously showed in [12] that the latter criteria for efficiency holds, in that for a graph of size n a step of an unbiased

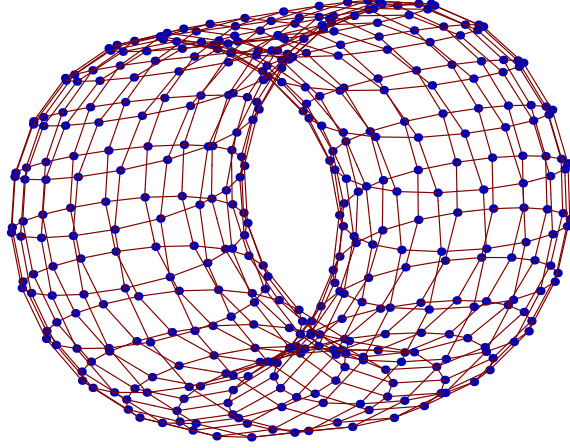


FIG. 5: A twisted toroid of dimension 20×20 .

walk requires $O(\log(n))$ elementary quantum gates. Fig. 6 extends this result, providing a circuit explicitly performing a step of a search for a marked node, given a twisted toroid of dimension $2^n \times 2^m$.

For this quantum search, the coin and shifting operator are again defined as in the hypercube example above, with the coin operator applying the Grover coin to every node except the marked node, on which the negative identity operator is applied. Each node has degree 4, and the ‘node’ states are represented by bit strings comprising the x- and y-coordinates of the grid from which the twisted toroid was constructed, such that the node (i, j) is represented by the bit string $(i - 1)$ concatenated with $(j - 1)$. Hence the shifting operator applied to node (i, j) can be implemented by either incrementing or decrementing the bit string associated with each of i and j (modulo 2^n and 2^m respectively), depending on the coin state of this node.

Incrementing and decrementing these bit strings is performed by the ‘incr’ and ‘decr’ operators in Fig. 6, each of which can be implemented using $O(n^2)$ 2-qubit gates via the circuits of Fig. 7. We are now in a position to calculate an upper bound on the number of 2-qubit gates required to implement Fig. 6. For a twisted toroid of dimension $2^n \times 2^m$, with $N = 2^n \times 2^m$ nodes, n ‘x-coord’ and m ‘y-coord’ qubits are required, together with 2 ‘subnode’ qubits. The controlled G and π gates are implementable using $O(n + m)$ 2-qubit gates, and since the increment and decrement gates on i qubits require $O(i^2)$ 2-qubit gates, the controlled increment and decrement operators are implementable using $O(n^2m + m^2n)$ 2-qubit gates. Note that since the degree of the graph is fixed at 4, only 2 subnode qubits

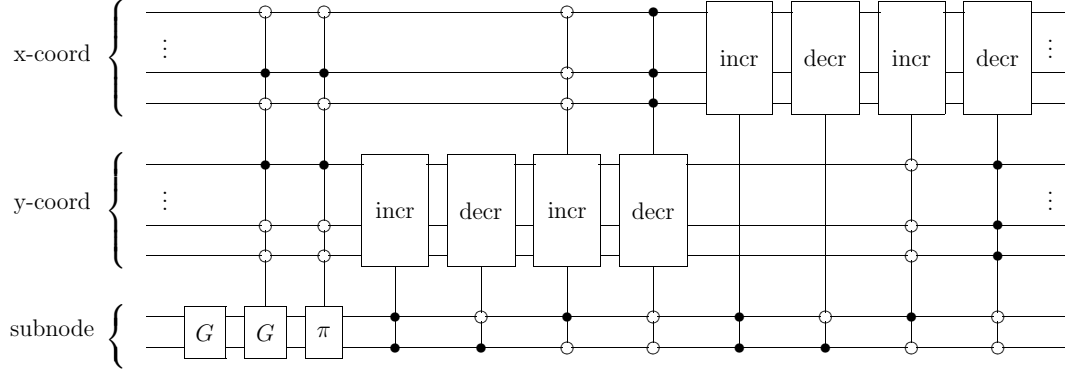


FIG. 6: Quantum circuit implementing a step of the quantum search along the twisted toroid of dimension $2^n \times 2^m$. The circuit contains n ‘x-coord’ qubits and n ‘y-coord’ qubits, and the marked node is represented by the state $|0 \cdots 10, 1 \cdots 00\rangle$.

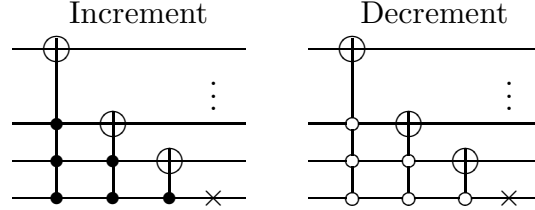


FIG. 7: Increment and decrement gates on n qubits, producing cyclic permutations of the 2^n bit string states. Each CNOT gate can be implemented using $O(i)$ 2-qubit gates, hence $O(n^2)$ 2-qubit gates are sufficient to implement the increment/decrement gates on n qubits.

are required regardless of the values of n and m . Hence a step of the quantum walk along a twisted toroid on $O(N)$ nodes can be implemented using $O(\log(N))$ 2-qubit gates.

It remains to show whether or not the circuit of Fig. 6 can perform an efficient search for the marked node - i.e. whether for a twisted toroid of dimension $n \times m$, the marked node can be found with $O(\sqrt{n} \times \sqrt{m})$ iterations of this circuit. To examine this, the behaviour of the walk was analyzed for a range of different toroid sizes. Regardless of the size, the search behaviour was in one respect successful, in that the probability to find the walk at the marked node reached a sufficiently high level after a certain number of steps, with fixed periodic behaviour. The approximate number of steps required was found for a range of twisted toroid sizes by calculating the period of the success probability (to find the walk at the marked node) with number of steps. The resulting correlation between toroid size (in number of vertices) and approximate number of steps required to reach maximum success

probability is shown in Fig. 8. These numerical results show that for a twisted toroid with N nodes, $O(\sqrt{N})$ steps of the walk are required to find the marked node, matching (up to a constant factor) the theoretical lower bound for unstructured search [28].

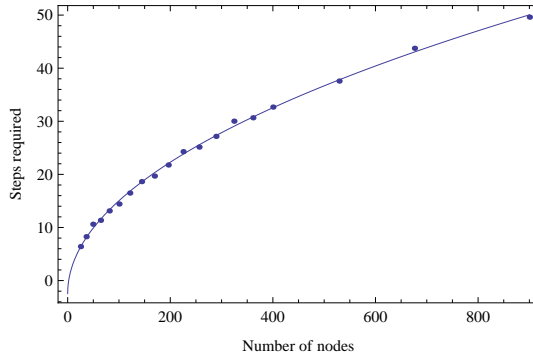


FIG. 8: Number of walking steps required to reach maximum success probability against the size of the graph. The solid line represents a \sqrt{N} fit to the numerical data.

Putting these results together, we see that this family of twisted toroids satisfy both criteria for efficient search via quantum walks, yielding for a graph on N nodes both an efficient implementation of the walks themselves (using $O(\log(N))$ 2-qubit gates), and requiring $O(\sqrt{N})$ steps of the walk to find a marked node with sufficiently high probability.

IV. CONCLUSIONS

The examples given in this paper are intended to provide a ‘proof of concept’, presenting graphs for which both efficient quantum search and efficient implementation of quantum walks are possible. By explicitly characterising the internal action of the oracle, these circuits provide a practically implementable method to experimentally test these quantum walk based search algorithms. In some previous studies, quantum speedups have been shown relative to oracles that cannot be efficiently implemented. Whilst these studies are very useful from a computational complexity viewpoint, the algorithms involved cannot be practically implemented.

-
- [1] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, in *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing* (ACM, New York, 2003), pp. 59–68.
 - [2] N. Shenvi, J. Kempe, and K. Whaley, *Phys. Rev. A* **67**, 052307 (2003).
 - [3] A. M. Childs and J. Goldstone, *Phys. Rev. A* **70**, 022314 (2004).
 - [4] A. M. Childs and J. Goldstone, *Phys. Rev. A* **70**, 042312 (2004).
 - [5] F. Magniez, A. Nayak, J. Roland, and M. Santha, in *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing* (ACM, New York, 2007), pp. 575–584.
 - [6] M. Santha, *LNCS* **4978**, 31 (2008).
 - [7] D. Reitzner, M. Hillery, E. Feldman, and V. Bužek, *Phys. Rev. A* **79**, 012323 (2009).
 - [8] C. Moore and A. Russell, in *RANDOM '02: Proceedings of the 6th International Workshop on Randomization and Approximation Techniques* (Springer-Verlag, London, UK, 2002), pp. 164–178.
 - [9] A. Ambainis, J. Kempe, and A. Rivosh, in *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms* (2005), p. 1099.
 - [10] V. Potoček, A. Gábris, T. Kiss, and I. Jex, *Phys. Rev. A* **79**, 012325 (2009).
 - [11] D. Aharonov and A. Ta-Shma, in *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing* (ACM, New York, 2003), pp. 20–29.
 - [12] B. L. Douglas and J. B. Wang, *Physical Review A* **79**, 052335 (2009).
 - [13] T. Loke and J. B. Wang, *Computer Physics Communications* **182**, 22852294 (2011).
 - [14] K. Manouchehri and J. Wang, *J. Phys. A* **41**, 065304 (2008).
 - [15] K. Manouchehri and J. B. Wang, *Physical Review A* **80**, 060304 (2009).
 - [16] B. Tregenna, W. Flanagan, R. Maile, and V. Kendon, *New J. Phys.* **5**, 83 (2003).
 - [17] D. Aharonov, A. Ambainis, J. Kempe, and U. V. Vazirani, in *ACM Symposium on Theory of Computing* (2001), pp. 50–59.
 - [18] E. Farhi and S. Gutmann, *Phys. Rev. A* **58**, 915 (1998).
 - [19] H. Gerhardt and J. Watrous, in *RANDOM-APPROX* (2003), pp. 290–301.
 - [20] J. Kempe, *Contemporary Physics* **44**, 307 (2003).
 - [21] A. Ambainis, in *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations*

- of Computer Science* (2004), pp. 22–31.
- [22] B. L. Douglas and J. B. Wang, *J. Phys. A: Math. Theor* **41**, 075303 (2008).
 - [23] S. D. Berry and J. B. Wang, *Physical Review A* **82**, 042333 (2010).
 - [24] S. D. Berry and J. B. Wang, *Physical Review A* **83**, 042317 (2011).
 - [25] S. Berry, P. Bourke, and J. B. Wang, *Computer Physics Communications* **182**, 22952302 (2011).
 - [26] Z. Diao, M. S. Zubairy, and G. Chen, *Z. Naturforsch* **57a**, 701 (2002).
 - [27] R. R. Tucci (2004), quant-ph/0407215.
 - [28] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, *SIAM Journal on Computing* **26**, 1510 (1997).

